A Guide to Preprocessing Calcium Imaging Movies

From calcium imaging movies to nwb file

28/01/2021

Introduction

This manual will help you carry out data analysis from 2-photon imaging movies, starting from the acquisition to the data analysis, focusing on data preprocessing. The manual is based on the combination of a few programming languages, different softwares and deep-learning algorithms allowing to preprocess the raw data, leading to its analysis. Each step will be detailed to ease this arduous process, especially, the preprocessing of the data.

However, prior to this process, some prerequisites need to be respected.

Prerequisites

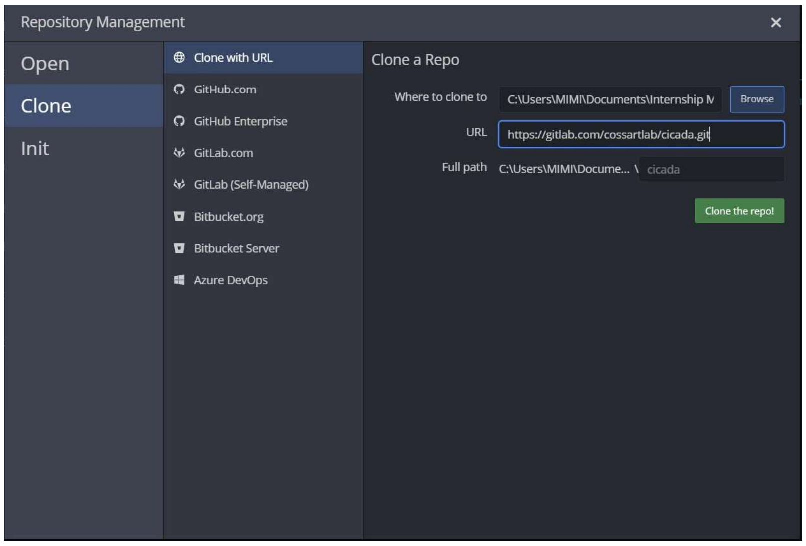
This part is dedicated to the installation of the different softwares and the potential issues that may occur during this step. Beware that some of the softwares needs to be installed in a certain order (it will be mentioned) so now let's dig in!

First, you need to install Anaconda. It is a free open-source distribution that manages packages for R and Python programming languages, the installation is simple: follow the steps at https://docs.anaconda.com/anaconda/install/ .

Now that the Python distribution is installed, you can install "Suite2p". Suite2p is an algorithm developed by Carsen Stringer and Marius Pachitariu for cell detection in two-photon calcium imaging movies. To install it, go on their github repository and follow the steps : https://github.com/MouseLand/suite2p . Note that, it is easier to copy/paste the environment.yml into a text file than cloning the repository. Also, keep in mind that the directory where you put the environment.yml is where your virtual environment is located i.e. where you will run suite2p. Beware that Pyqt (a Python package for Graphic Interface) won't work on Yosemite Mac OS, so you won't be able to install suite2p…

The second software to install is Gitkraken, a Graphic User Interface for Git. It will help you clone repositories, commit, drag/drop, track your tasks, projects on gitlab or github. To install it, follow the steps on https://www.gitkraken.com/download.

Once Gitkraken is installed, create an account and you will be ready to clone repositories. The screenshot below shows you how to do it. For the first repository, create a file “deepcinac_gitlab” and clone it at https://gitlab.com/cossartlab/deepcinac.git and put it in the “deepcinac_gitlab” file . For the second one, create another file “cicada_gitlab” and clone the CICADA repos at https://gitlab.com/cossartlab/cicada.git and put it in the “cicada_gitlab” file.

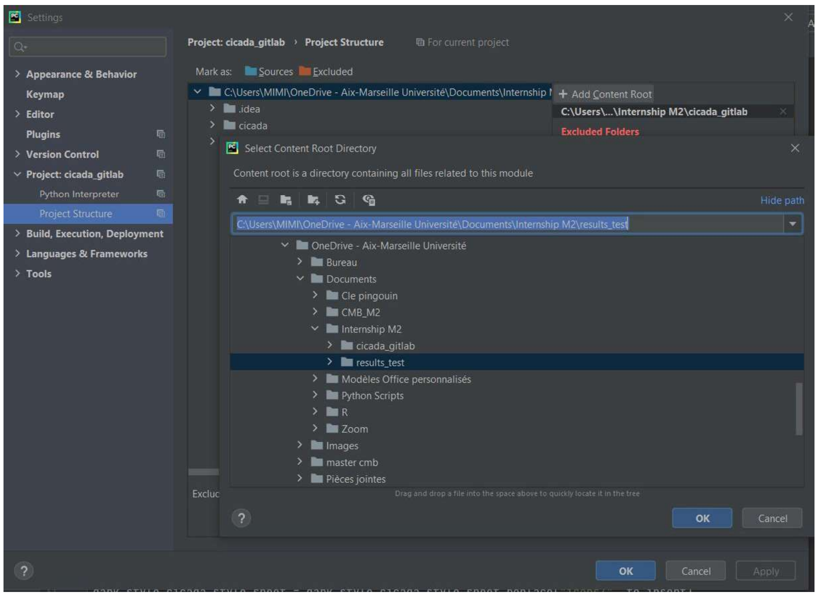


Where to clone to C:\Users\MIMI\Documents\Internship M

URL https://gitlab.com/cossartlab/cicada.git

Full path C:\Users\MIMI\Docume... \ cicada

The next step is the installation of Pycharm Community that you can find on https://www.jetbrains.com/fr-fr/pycharm/download/#section=windows and choose the 3.3 version. (more details on Pycharm IDE in the pptx related). Now that you have installed it, you have to add your project to the path, in other words, add the folder containing all the files related to the module. For that, go to `File > Settings > Project:Cicada_gitlab > project structure > add content root`, like below:



Select Content Root Directory

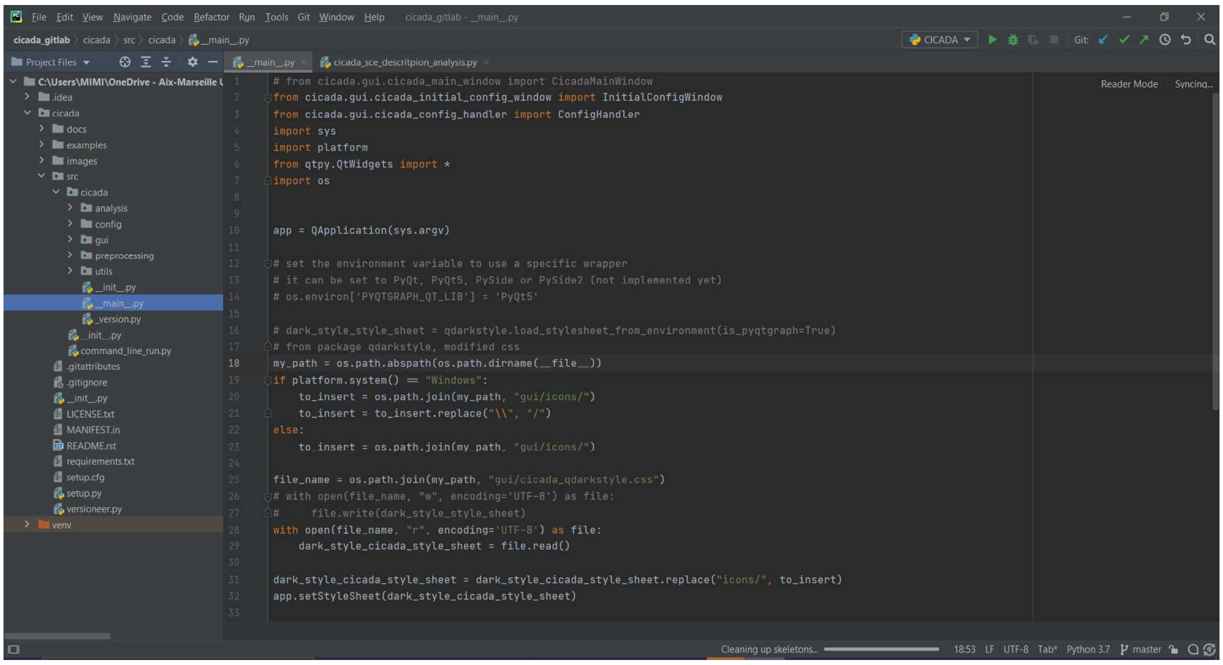
Content root is a directory containing all files related to this module

C:\Users\MIMI\OneDrive - Aix-Marseille Université\Documents\Internship M2\results_test

Drag and drop a file into the space above to quickly locate it in the tree

To ensure that CICADA is working well, open Pycharm and `Open Project`, go to the directory where cicada gitlab repository is located and select it. You will have the arborescence of the folders and subfolders of the project on the left.

Select in the arborescence `cicada_gitlab>cicada>src>cicada>main.py` , the script will appear in the right panel. Change the configuration to `CICADA` and run the script.

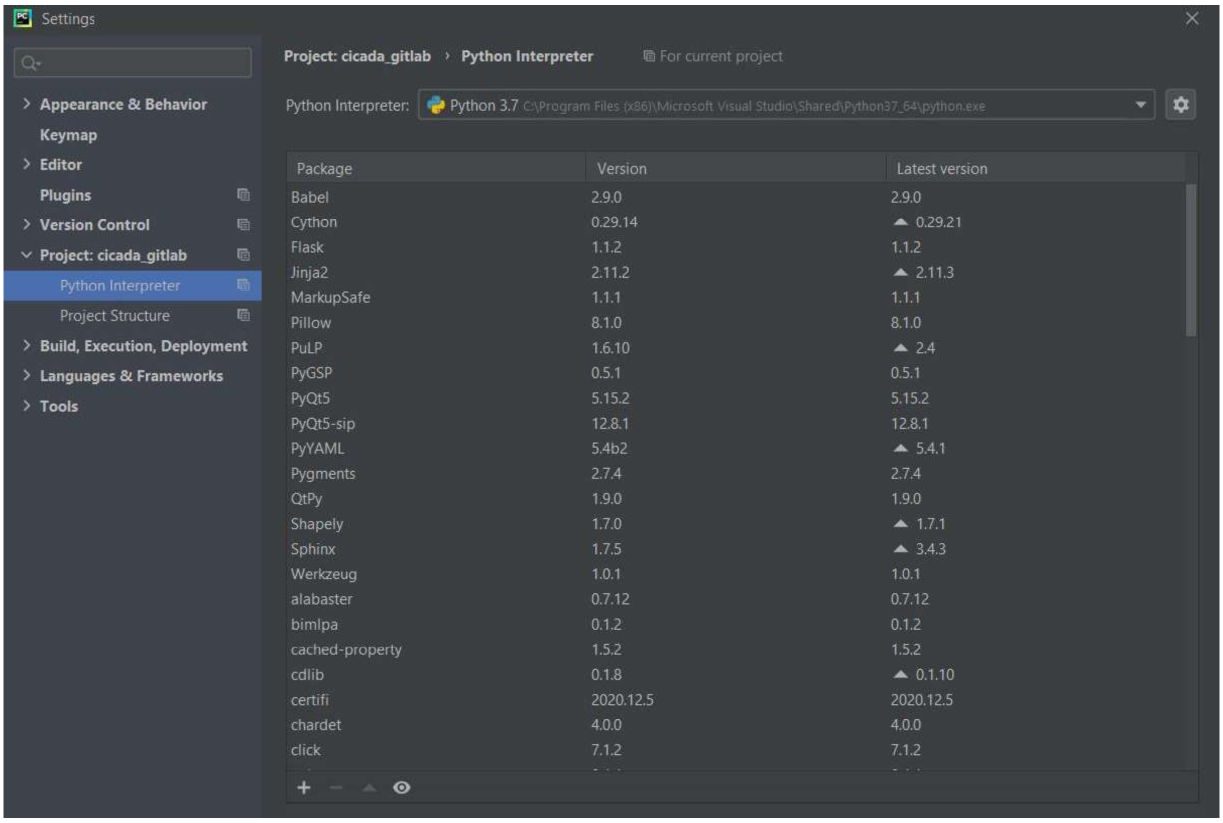


```python
# from cicada.gui.cicada_main_window import CicadaMainWindow
from cicada.gui.cicada_initial_config_window import InitialConfigWindow
from cicada.gui.cicada_config_handler import ConfigHandler
import sys
import platform
from qtpy.QtWidgets import *
import os


app = QApplication(sys.argv)

# set the environment variable to use a specific wrapper
# it can be set to PyQt, PyQt5, PySide or PySide2 (not implemented yet)
# os.environ['PYQTGRAPH_QT_LIB'] = 'PyQt5'

# dark_style_style_sheet = qdarkstyle.load_stylesheet_from_environment(is_pyqtgraph=True)
# from package qdarkstyle, modified css
my_path = os.path.abspath(os.path.dirname(__file__))
if platform.system() == "Windows":
    to_insert = os.path.join(my_path, "gui/icons/")
    to_insert = to_insert.replace("\\", "/")
else:
    to_insert = os.path.join(my_path, "gui/icons/")

file_name = os.path.join(my_path, "gui/cicada_qdarkstyle.css")
# with open(file_name, "w", encoding='UTF-8') as file:
#     file.write(dark_style_style_sheet)
with open(file_name, "r", encoding='UTF-8') as file:
    dark_style_cicada_style_sheet = file.read()

dark_style_cicada_style_sheet = dark_style_cicada_style_sheet.replace("icons/", to_insert)
app.setStyleSheet(dark_style_cicada_style_sheet)
```

At this level, beware of the compatibility and potential conflicts with the packages you will need to install. You will find below a list of the packages that I personally installed with a Windows 10 and a Pycharm 3.3 with a Python interpreter 3.7.

Tip: if you don’t know which interpreter you use on Pycharm, go in

`File > Settings > Project:Cicada_gitlab > python interpreter`. This also allows you to see the list of the installed packages for each interpreter and manage their version.



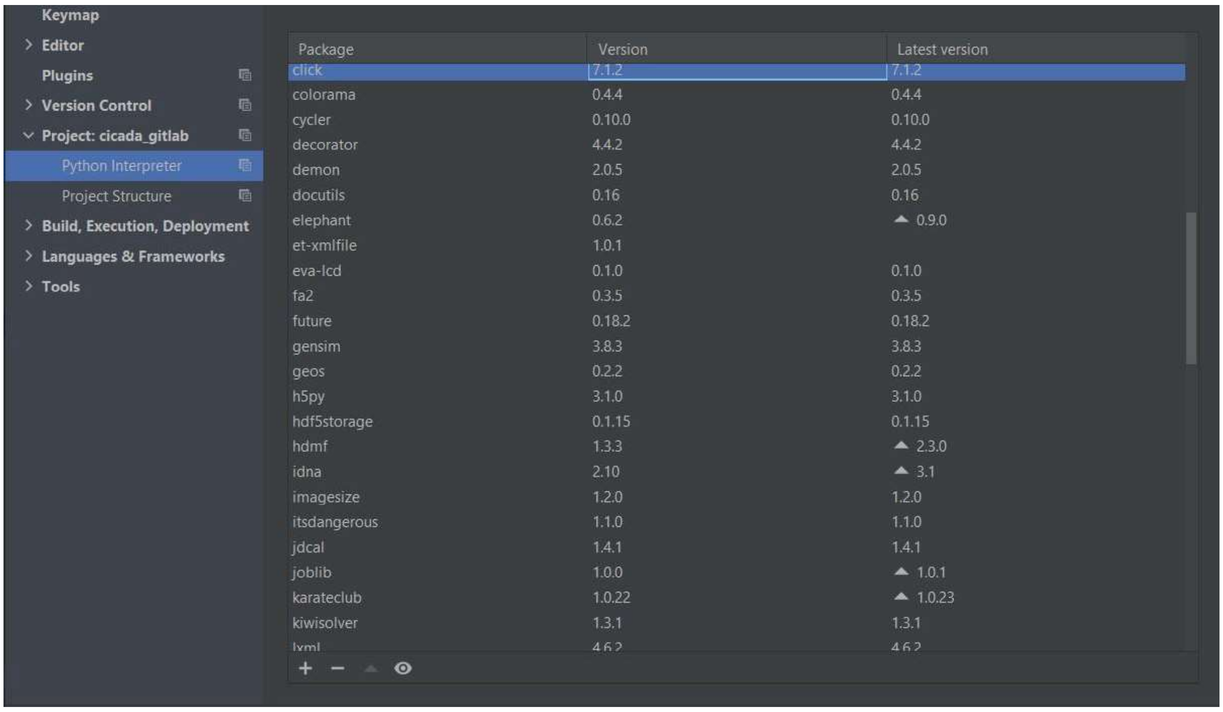
Settings

- Appearance & Behavior
- Keymap
- Editor
- Plugins
- Version Control
- Project: cicada_gitlab
  - Python Interpreter
  - Project Structure
- Build, Execution, Deployment
- Languages & Frameworks
- Tools

Project: cicada_gitlab › Python Interpreter For current project

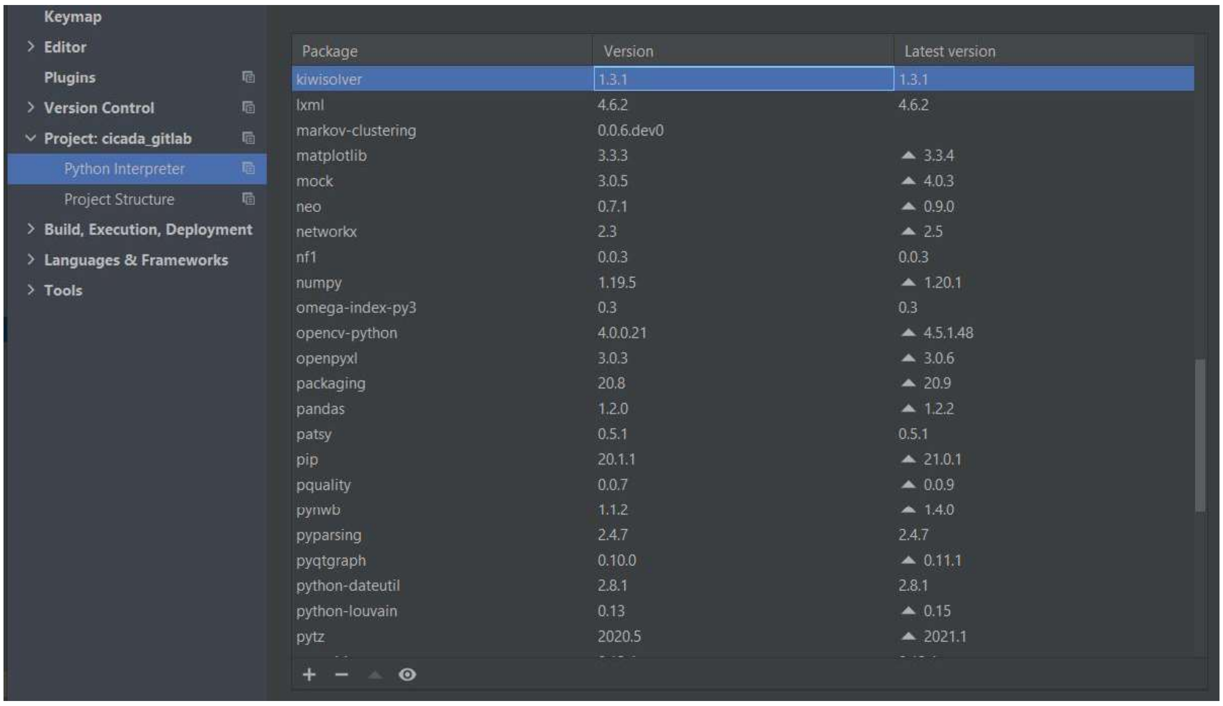
Python Interpreter: Python 3.7 C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python37_64\python.exe

Package	Version	Latest version
Babel	2.9.0	2.9.0
Cython	0.29.14	▲ 0.29.21
Flask	1.1.2	1.1.2
Jinja2	2.11.2	▲ 2.11.3
MarkupSafe	1.1.1	1.1.1
Pillow	8.1.0	8.1.0
PuLP	1.6.10	▲ 2.4
PyGSP	0.5.1	0.5.1
PyQt5	5.15.2	5.15.2
PyQt5-sip	12.8.1	12.8.1
PyYAML	5.4b2	▲ 5.4.1
Pygments	2.7.4	2.7.4
QtPy	1.9.0	1.9.0
Shapely	1.7.0	▲ 1.7.1
Sphinx	1.7.5	▲ 3.4.3
Werkzeug	1.0.1	1.0.1
alabaster	0.7.12	0.7.12
bimlpa	0.1.2	0.1.2
cached-property	1.5.2	1.5.2
cdlib	0.1.8	▲ 0.1.10
certifi	2020.12.5	2020.12.5
chardet	4.0.0	4.0.0
click	7.1.2	7.1.2

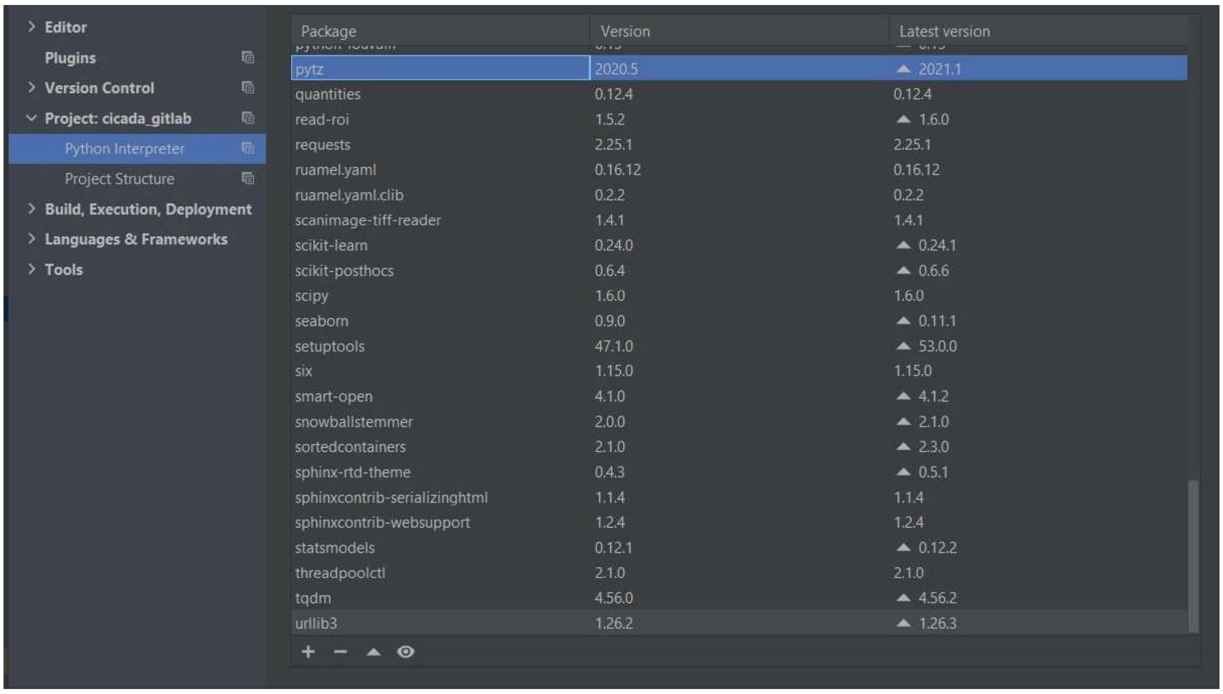


- Keymap
- Editor
- Plugins
- Version Control
- Project: cicada_gitlab
  - Python Interpreter
  - Project Structure
- Build, Execution, Deployment
- Languages & Frameworks
- Tools

Package	Version	Latest version
click	7.1.2	7.1.2
colorama	0.4.4	0.4.4
cycler	0.10.0	0.10.0
decorator	4.4.2	4.4.2
demon	2.0.5	2.0.5
docutils	0.16	0.16
elephant	0.6.2	▲ 0.9.0
et-xmlfile	1.0.1	
eva-lcd	0.1.0	0.1.0
fa2	0.3.5	0.3.5
future	0.18.2	0.18.2
gensim	3.8.3	3.8.3
geos	0.2.2	0.2.2
h5py	3.1.0	3.1.0
hdf5storage	0.1.15	0.1.15
hdmf	1.3.3	▲ 2.3.0
idna	2.10	▲ 3.1
imagesize	1.2.0	1.2.0
itsdangerous	1.1.0	1.1.0
jdcal	1.4.1	1.4.1
joblib	1.0.0	▲ 1.0.1
karateclub	1.0.22	▲ 1.0.23
kiwisolver	1.3.1	1.3.1
lxml	4.6.2	4.6.2

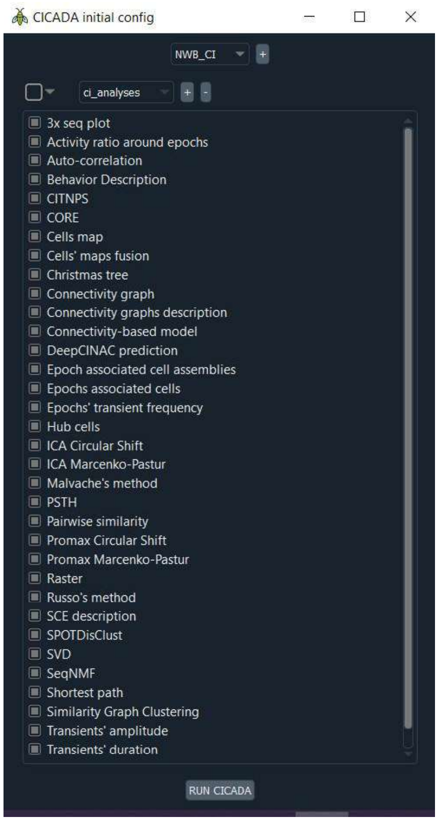


Package	Version	Latest version
kiwisolver	1.3.1	1.3.1
lxml	4.6.2	4.6.2
markov-clustering	0.0.6.dev0	
matplotlib	3.3.3	▲ 3.3.4
mock	3.0.5	▲ 4.0.3
neo	0.7.1	▲ 0.9.0
networkx	2.3	▲ 2.5
nf1	0.0.3	0.0.3
numpy	1.19.5	▲ 1.20.1
omega-index-py3	0.3	0.3
opencv-python	4.0.0.21	▲ 4.5.1.48
openpyxl	3.0.3	▲ 3.0.6
packaging	20.8	▲ 20.9
pandas	1.2.0	▲ 1.2.2
patsy	0.5.1	0.5.1
pip	20.1.1	▲ 21.0.1
pquality	0.0.7	▲ 0.0.9
pynwb	1.1.2	▲ 1.4.0
pyparsing	2.4.7	2.4.7
pyqtgraph	0.10.0	▲ 0.11.1
python-dateutil	2.8.1	2.8.1
python-louvain	0.13	▲ 0.15
pytz	2020.5	▲ 2021.1



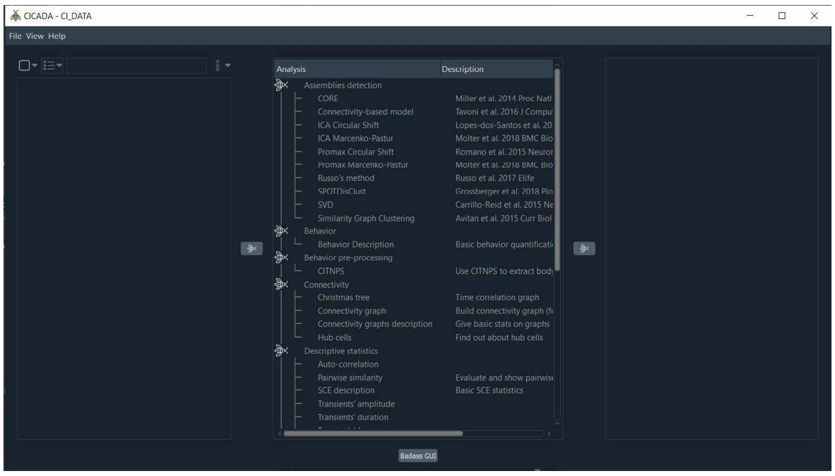
Package	Version	Latest version
pytz	2020.5	▲ 2021.1
quantities	0.12.4	0.12.4
read-roi	1.5.2	▲ 1.6.0
requests	2.25.1	2.25.1
ruamel.yaml	0.16.12	0.16.12
ruamel.yaml.clib	0.2.2	0.2.2
scanimage-tiff-reader	1.4.1	1.4.1
scikit-learn	0.24.0	▲ 0.24.1
scikit-posthocs	0.6.4	▲ 0.6.6
scipy	1.6.0	1.6.0
seaborn	0.9.0	▲ 0.11.1
setuptools	47.1.0	▲ 53.0.0
six	1.15.0	1.15.0
smart-open	4.1.0	▲ 4.1.2
snowballstemmer	2.0.0	▲ 2.1.0
sortedcontainers	2.1.0	▲ 2.3.0
sphinx-rtd-theme	0.4.3	▲ 0.5.1
sphinxcontrib-serializinghtml	1.1.4	1.1.4
sphinxcontrib-websupport	1.2.4	1.2.4
statsmodels	0.12.1	▲ 0.12.2
threadpoolctl	2.1.0	2.1.0
tqdm	4.56.0	▲ 4.56.2
urllib3	1.26.2	▲ 1.26.3

Once the packages are installed correctly, you will be able to run and see the initial configuration interface of CICADA (figure below). This panel summarizes the operations that you can perform with CICADA.



- 3x seq plot
- Activity ratio around epochs
- Auto-correlation
- Behavior Description
- CITNPS
- CORE
- Cells map
- Cells' maps fusion
- Christmas tree
- Connectivity graph
- Connectivity graphs description
- Connectivity-based model
- DeepCINAC prediction
- Epoch associated cell assemblies
- Epochs associated cells
- Epochs' transient frequency
- Hub cells
- ICA Circular Shift
- ICA Marcenko-Pastur
- Malvache's method
- PSTH
- Pairwise similarity
- Promax Circular Shift
- Promax Marcenko-Pastur
- Raster
- Russo's method
- SCE description
- SPOTDisClust
- SVD
- SeqNMF
- Shortest path
- Similarity Graph Clustering
- Transients' amplitude
- Transients' duration

Hit the `RUN CICADA` button. The next window has 3 panels: left one is for the chosen .nwb files to process (just select the directory), the center is to select the wanted tasks and the right one is for results and tasks tracking that Cicada is performing. Many tasks are available, you just have to select it on the center panel, before launching the Badass GUI. Once you hit the `Badass GUI` button, another window corresponding to the task you want appears.



Analysis	Description
Assemblies detection	
CORE	Miller et al. 2014 Proc Natl
Connectivity-based model	Tavoni et al. 2016 J Compu
ICA Circular Shift	Lopes-dos-Santos et al. 20
ICA Marcenko-Pastur	Molter et al. 2018 BMC Bio
Promax Circular Shift	Romano et al. 2015 Neuror
Promax Marcenko-Pastur	Molter et al. 2018 BMC Bio
Russo's method	Russo et al. 2017 Elife
SPOTDisClust	Grossberger et al. 2018 Plo
SVD	Carrillo-Reid et al. 2015 Ne
Similarity Graph Clustering	Avitan et al. 2015 Curr Biol
Behavior	
Behavior Description	Basic behavior quantificati
Behavior pre-processing	
CITNPS	Use CITNPS to extract body
Connectivity	
Christmas tree	Time correlation graph
Connectivity graph	Build connectivity graph (f
Connectivity graphs description	Give basic stats on graphs
Hub cells	Find out about hub cells
Descriptive statistics	
Auto-correlation	
Pairwise similarity	Evaluate and show pairwis
SCE description	Basic SCE statistics
Transients' amplitude	
Transients' duration	

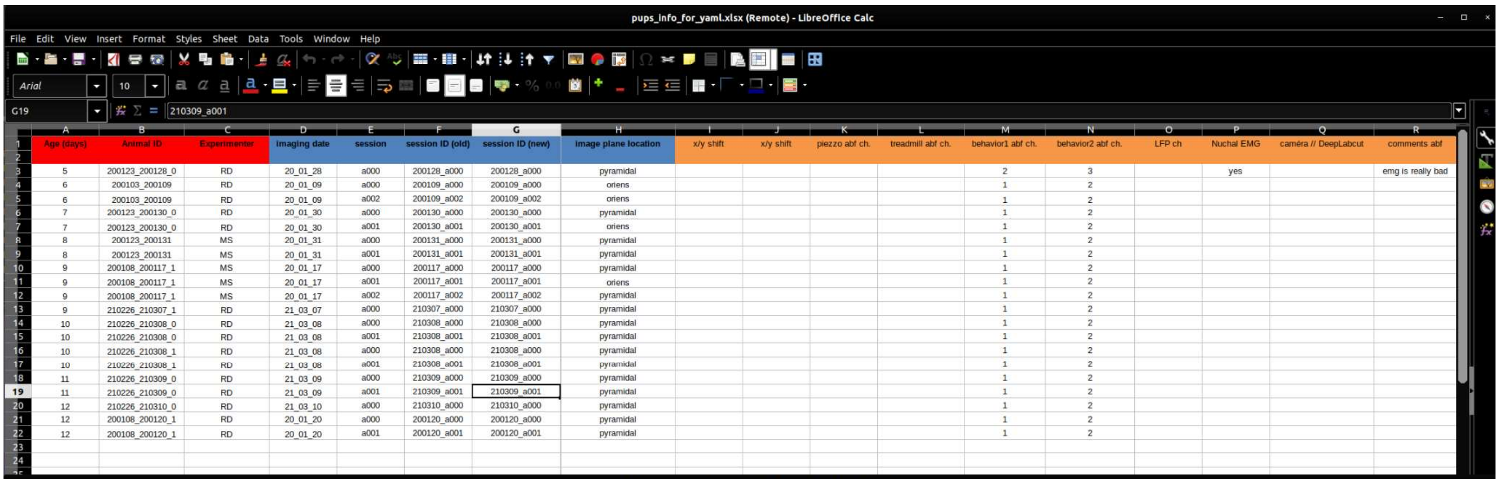
For some further preprocessing steps, you will need to install a computing environment focused on mathematical and technical computing, the best in the field is Matlab. Assuming that you can pay the mensual subscription to Matlab or have a licence thanks to your university, download the version 2018a on: https://fr.mathworks.com/products/get-matlab.html?s_tid=gn_getml .

Creation and implementation of the architecture

Data preprocessing and analysis are long and recurrent tasks. To optimize your time and not get bogged down in an ocean of data, you will need to implement a clear and well-ordered architecture to organize your data.

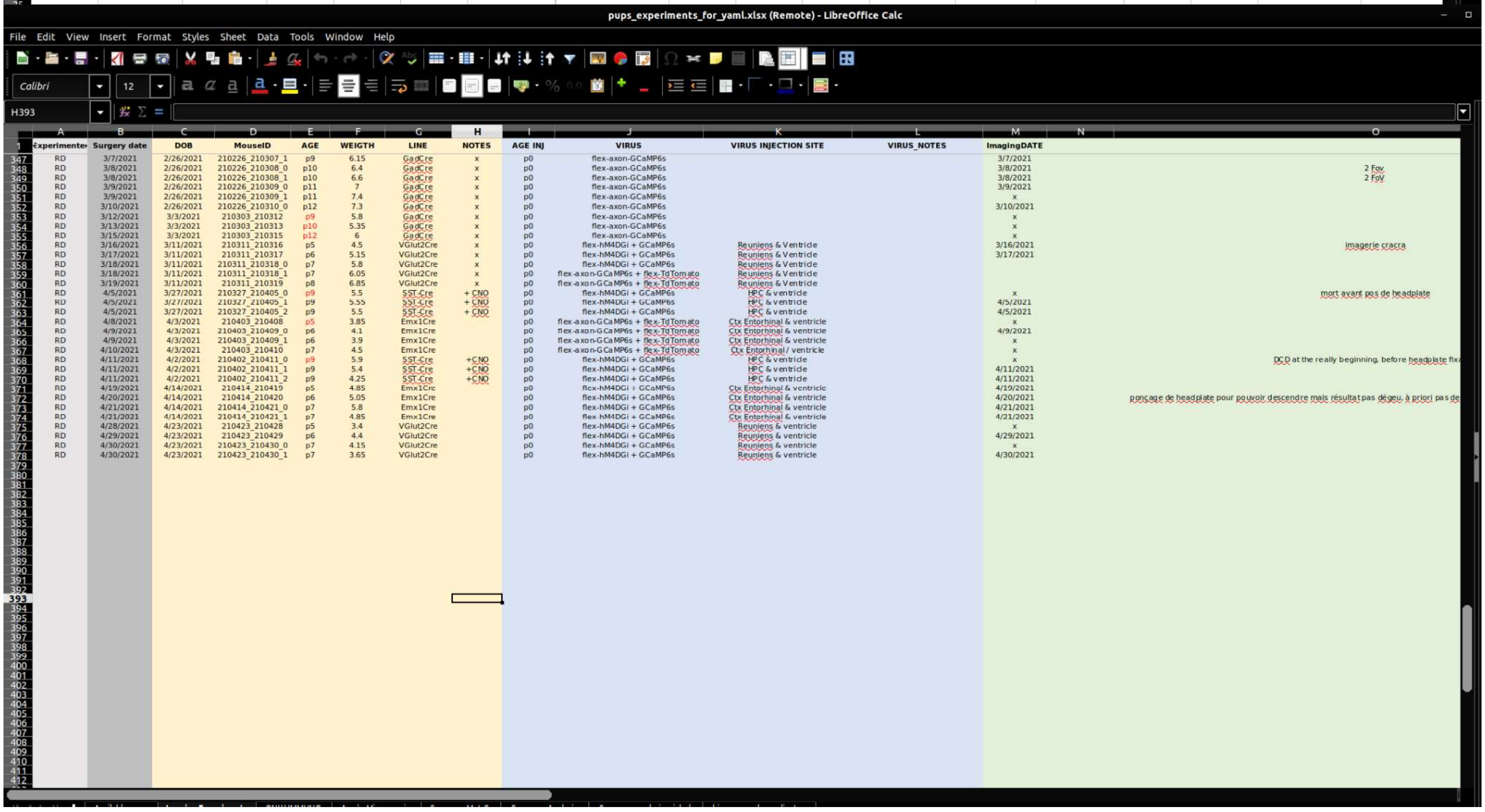
First, a unique ID is assigned to each experiment (i.e. each imaging session), its pattern is as follows: "*birthdate_surgerydate_imagingdate_fieldname*". For example, the imaging movie of a mouse born on 14/10/2021, underwent surgery on 22/01/2021 and imaged one hour after, is named *210114_210122_210122_a000*. Note that "fieldname" is a000, since it is the first session on the mouse, however if you want to image another field, then the imaging movie will be named *210114_210122_210122_a001*.

Then, for each experiment, experimental metadatas are stored in two excel files. To be more efficient, use the existing excel files, on the NAS, located in the "new_yaml_files" folder (just make sure you don't erase the others' lines). They are called *pups_experiments_for_yaml*, referred to as **main** and *pups_info_for_yaml,* referred to as the **external**. The **main** contains many sheets, including the "*Imaging*" sheet in which the information concerning the pups, and the imaging field and observations, are. The **external** one contains many sheets as well, these sheets describe the kind of experiments that were conducted, for example, the “SST-CRE“ sheet. In this external file, there are details of the weight and the age of the pup, the drug administered and experimental conditions. For both files, each line describes one experiment (→ one pup can refer to multiple lines).


pups_info_for_yaml.xlsx (Remote) - LibreOffice Calc

G19 | 210309_a001

Age (days)	Animal ID	Experimenter	imaging date	session	session ID (old)	session ID (new)	image plane location	x/y shift	x/y shift	piezzo abf ch.	treadmill abf ch.	behavior1 abf ch.	behavior2 abf ch.	LFP ch	Nuchal EMG	caméra // DeepLabcut	comments abf
5	200123_200128_0	RD	20_01_28	a000	200128_a000	200128_a000	pyramidal					2	3		yes		emg is really bad
6	200103_200109	RD	20_01_09	a000	200109_a000	200109_a000	oriens					1	2				
6	200103_200109	RD	20_01_09	a002	200109_a002	200109_a002	oriens					1	2				
7	200123_200130_0	RD	20_01_30	a000	200130_a000	200130_a000	pyramidal					1	2				
7	200123_200130_0	RD	20_01_30	a001	200130_a001	200130_a001	oriens					1	2				
8	200123_200131	MS	20_01_31	a000	200131_a000	200131_a000	pyramidal					1	2				
8	200123_200131	MS	20_01_31	a001	200131_a001	200131_a001	pyramidal					1	2				
9	200108_200117_1	MS	20_01_17	a000	200117_a000	200117_a000	pyramidal					1	2				
9	200108_200117_1	MS	20_01_17	a001	200117_a001	200117_a001	oriens					1	2				
9	200108_200117_1	MS	20_01_17	a002	200117_a002	200117_a002	pyramidal					1	2				
9	210226_210307_1	RD	21_03_07	a000	210307_a000	210307_a000	pyramidal					1	2				
10	210226_210308_0	RD	21_03_08	a000	210308_a000	210308_a000	pyramidal					1	2				
10	210226_210308_0	RD	21_03_08	a001	210308_a001	210308_a001	pyramidal					1	2				
10	210226_210308_1	RD	21_03_08	a000	210308_a000	210308_a000	pyramidal					1	2				
10	210226_210308_1	RD	21_03_08	a001	210308_a001	210308_a001	pyramidal					1	2				
11	210226_210309_0	RD	21_03_09	a000	210309_a000	210309_a000	pyramidal					1	2				
11	210226_210309_0	RD	21_03_09	a001	210309_a001	210309_a001	pyramidal					1	2				
12	210226_210310_0	RD	21_03_10	a000	210310_a000	210310_a000	pyramidal					1	2				
12	200108_200120_1	RD	20_01_20	a000	200120_a000	200120_a000	pyramidal					1	2				
12	200108_200120_1	RD	20_01_20	a001	200120_a001	200120_a001	pyramidal					1	2				


pups_experiments_for_yaml.xlsx (Remote) - LibreOffice Calc

H393

Experimenter	Surgery date	DOB	MouseID	AGE	WEIGTH	LINE	NOTES	AGE INJ	VIRUS	VIRUS INJECTION SITE	VIRUS_NOTES	ImagingDATE		
RD	3/7/2021	2/26/2021	210226_210307_1	p9	6.15	GadCre	x	p0	flex-axon-GCaMP6s			3/7/2021		
RD	3/8/2021	2/26/2021	210226_210308_0	p10	6.4	GadCre	x	p0	flex-axon-GCaMP6s			3/8/2021		2 Fov
RD	3/8/2021	2/26/2021	210226_210308_1	p10	6.6	GadCre	x	p0	flex-axon-GCaMP6s			3/8/2021		2 FoV
RD	3/9/2021	2/26/2021	210226_210309_0	p11	7	GadCre	x	p0	flex-axon-GCaMP6s			3/9/2021		
RD	3/9/2021	2/26/2021	210226_210309_1	p11	7.4	GadCre	x	p0	flex-axon-GCaMP6s			x		
RD	3/10/2021	2/26/2021	210226_210310_0	p12	7.3	GadCre	x	p0	flex-axon-GCaMP6s			3/10/2021		
RD	3/12/2021	3/3/2021	210303_210312	p9	5.8	GadCre	x	p0	flex-axon-GCaMP6s			x		
RD	3/13/2021	3/3/2021	210303_210313	p10	5.35	GadCre	x	p0	flex-axon-GCaMP6s			x		
RD	3/15/2021	3/3/2021	210303_210315	p12	6	GadCre	x	p0	flex-axon-GCaMP6s			x		
RD	3/16/2021	3/11/2021	210311_210316	p5	4.5	VGlut2Cre	x	p0	flex-hM4DGi + GCaMP6s	Reuniens & Ventricle		3/16/2021		imagerie cracra
RD	3/17/2021	3/11/2021	210311_210317	p6	5.15	VGlut2Cre	x	p0	flex-hM4DGi + GCaMP6s	Reuniens & Ventricle		3/17/2021		
RD	3/18/2021	3/11/2021	210311_210318_0	p7	5.8	VGlut2Cre	x	p0	flex-hM4DGi + GCaMP6s	Reuniens & Ventricle				
RD	3/18/2021	3/11/2021	210311_210318_1	p7	6.05	VGlut2Cre	x	p0	flex-axon-GCaMP6s + flex-TdTomato	Reuniens & Ventricle				
RD	3/19/2021	3/11/2021	210311_210319	p8	6.85	VGlut2Cre	x	p0	flex-axon-GCaMP6s + flex-TdTomato	Reuniens & Ventricle				
RD	4/5/2021	3/27/2021	210327_210405_0	p9	5.5	SST-Cre	+ CNO	p0	flex-hM4DGi + GCaMP6s	HPC & ventricle		x		mort avant pos de headplate
RD	4/5/2021	3/27/2021	210327_210405_1	p9	5.55	SST-Cre	+ CNO	p0	flex-hM4DGi + GCaMP6s	HPC & ventricle		4/5/2021		
RD	4/5/2021	3/27/2021	210327_210405_2	p9	5.5	SST-Cre	+ CNO	p0	flex-hM4DGi + GCaMP6s	HPC & ventricle		4/5/2021		
RD	4/8/2021	4/3/2021	210403_210408	p5	3.85	Emx1Cre		p0	flex-axon-GCaMP6s + flex-TdTomato	Ctx Entorhinal & ventricle		x		
RD	4/9/2021	4/3/2021	210403_210409_0	p6	4.1	Emx1Cre		p0	flex-axon-GCaMP6s + flex-TdTomato	Ctx Entorhinal & ventricle		4/9/2021		
RD	4/9/2021	4/3/2021	210403_210409_1	p6	3.9	Emx1Cre		p0	flex-axon-GCaMP6s + flex-TdTomato	Ctx Entorhinal & ventricle		x		
RD	4/10/2021	4/3/2021	210403_210410	p7	4.5	Emx1Cre		p0	flex-axon-GCaMP6s + flex-TdTomato	Ctx Entorhinal / ventricle		x		
RD	4/11/2021	4/2/2021	210402_210411_0	p9	5.9	SST-Cre	+CNO	p0	flex-hM4DGi + GCaMP6s	HPC & ventricle		x		DCD at the really beginning, before headplate fix
RD	4/11/2021	4/2/2021	210402_210411_1	p9	5.4	SST-Cre	+CNO	p0	flex-hM4DGi + GCaMP6s	HPC & ventricle		4/11/2021		
RD	4/11/2021	4/2/2021	210402_210411_2	p9	4.25	SST-Cre	+CNO	p0	flex-hM4DGi + GCaMP6s	HPC & ventricle		4/11/2021		
RD	4/19/2021	4/14/2021	210414_210419	p5	4.85	Emx1Cre		p0	flex-hM4DGi + GCaMP6s	Ctx Entorhinal & ventricle		4/19/2021		
RD	4/20/2021	4/14/2021	210414_210420	p6	5.05	Emx1Cre		p0	flex-hM4DGi + GCaMP6s	Ctx Entorhinal & ventricle		4/20/2021	ponçage de headplate pour pouvoir descendre mais résultat pas dégeu, à priori pas de	
RD	4/21/2021	4/14/2021	210414_210421_0	p7	5.8	Emx1Cre		p0	flex-hM4DGi + GCaMP6s	Ctx Entorhinal & ventricle		4/21/2021		
RD	4/21/2021	4/14/2021	210414_210421_1	p7	4.85	Emx1Cre		p0	flex-hM4DGi + GCaMP6s	Ctx Entorhinal & ventricle		4/21/2021		
RD	4/28/2021	4/23/2021	210423_210428	p5	3.4	VGlut2Cre		p0	flex-hM4DGi + GCaMP6s	Reuniens & ventricle		x		
RD	4/29/2021	4/23/2021	210423_210429	p6	4.4	VGlut2Cre		p0	flex-hM4DGi + GCaMP6s	Reuniens & ventricle		4/29/2021		
RD	4/30/2021	4/23/2021	210423_210430_0	p7	4.15	VGlut2Cre		p0	flex-hM4DGi + GCaMP6s	Reuniens & ventricle		x		
RD	4/30/2021	4/23/2021	210423_210430_1	p7	3.65	VGlut2Cre		p0	flex-hM4DGi + GCaMP6s	Reuniens & ventricle		4/30/2021		

Imaging and Acquisition

The imaging session is performed one hour after the surgery, in the casquette. The cannula is filled with a liquid since we use immersion microscopy. Each calcium imaging session is composed of 5 blocks of 2500 frames in .tiff format. In our case, we also recorded simultaneously the behavior of the mouse with two cameras, the two movies were in tiff format. Once the acquisition is launched, the signals from the two cameras synchronized with the microscope's are recorded with Clampex and exported as a .abf file.

Once the imaging is over, convert the .msr files from ImspectorPro in .tiff, for that, in ImspectorPro, go in File>Batch conversion, select your directory and the .msr files to convert. When you launch the conversion, it is common that ImspectorPro is lagging.

We also need to convert the behavior movies in .avi. The conversion is performed by running the code 'convert_tiff_to_avi' on pycharm. In the script, change the name of the experiment: "subject_id", the name of the camera (cam1/cam2): "cam_folder_id_1" and the name of the field: "cam_folder_id_2". Pycharm allows you to perform simultaneous conversions, indeed, you have a pin icon, on the lower left of the console. Pin the running tab and open another one to run another conversion. That way, you can run up to 5 conversions at the same time. However, this step is a bit time-consuming and uses a lot of CPU, therefore, it is preferred to run conversions at night.

The following steps don't require the immediate use of the converted movies. At this point, the raw data consists of the 5 tiffs and the .abf file. The next step is the concatenation of the 5 .tiff movies into a final one. You can perform it on FIJI (use concatenate in Tools menu). Now that we have an unique imaging movie and the file containing the synchronized data, we can move to the preprocessing.

Main steps of the preprocessing

Convert behavior monitoring from “.tiff” to “.avi”

To make sure the conversion can be done without changing the python code this arborescence should be respected:

- Animal_ID (*e.g.* folder name “p5_210105_210110”)
    - cam_ID (folder name is either “cam1” or “cam2”)
        - field_of_view_ID (*e.g.* folder name “210110_a000)

The conversion should be done directly on the acquisition computer to save time. It can also be done on another computer if the data is on an external hard drive(just take more time)

Use the code: only a few lines have to be modified to set the paths.

Here add some screenshots showing how to adapt the paths to convert the movies

Data transfer on local disk

- Create a folder that will be used later to create the “.nwb” data file.
- Transfer data from the disk extension to the local disk in the “nwb” creation folder:

→ “.abf” data from the axoscope recording (be sure it is recorded @50 kHz)

→ “.avi” data from the behavior monitoring

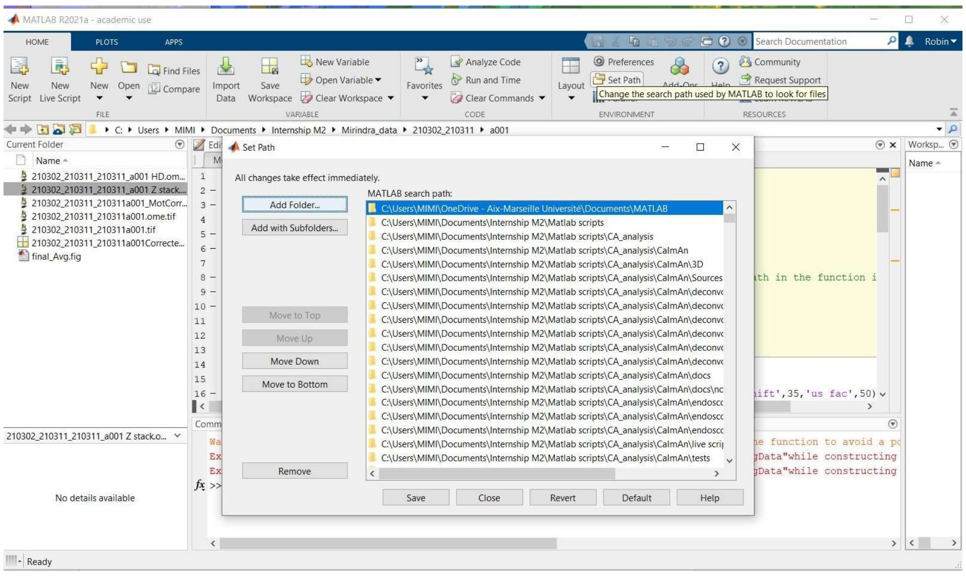
→ “.tiff” files

Motion correction

During the imaging, distortions following x,y-axis and z-axis are likely to occur. The motion correction corrects these distortions as much as possible and relies on a simple principle: a frame (or a frame resulting from the mean of stable frames) is used as reference and each frame is aligned with it. There are two methods available: the cropping method by M. Picardo and the black stripes by Yannick.

- The cropping method involves the cropping, for all the frames, of the “unaligned parts” for each frame, in order to equalize. This method offers a great deal of the distortion in the x and y axis.
- The back stripes consist of putting black stripes on unaligned parts. This method deals averagely with distortion in x and y axis; however, it behaves greatly with any offsetting in z-axis.

Whatever the method you choose, you will run the motion correction on Matlab. They are named **MotCorre.m** and **Yannick_MotC.m** . For both, you have to change the directory to where your raw movie is located. You can change it either from the left panel or add the folder containing the tiff, using "set path" in the Home panel.



C: ▸ Users ▸ MIMI ▸ Documents ▸ Internship M2 ▸ Mirindra_data ▸ 210302_210311 ▸ a001

Current Folder

210302_210311_210311_a001 HD.om...
210302_210311_210311_a001 Z stack...
210302_210311_210311a001_MotCorr...
210302_210311_210311a001.ome.tif
210302_210311_210311a001.tif
210302_210311_210311a001Correcte...
final_Avg.fig

No details available

All changes take effect immediately.

MATLAB search path:

C:\Users\MIMI\OneDrive - Aix-Marseille Université\Documents\MATLAB
C:\Users\MIMI\Documents\Internship M2\Matlab scripts
C:\Users\MIMI\Documents\Internship M2\Matlab scripts\CA_analysis
C:\Users\MIMI\Documents\Internship M2\Matlab scripts\CA_analysis\CalmAn
C:\Users\MIMI\Documents\Internship M2\Matlab scripts\CA_analysis\CalmAn\3D
C:\Users\MIMI\Documents\Internship M2\Matlab scripts\CA_analysis\CalmAn\Sources
C:\Users\MIMI\Documents\Internship M2\Matlab scripts\CA_analysis\CalmAn\deconv
C:\Users\MIMI\Documents\Internship M2\Matlab scripts\CA_analysis\CalmAn\deconv
C:\Users\MIMI\Documents\Internship M2\Matlab scripts\CA_analysis\CalmAn\deconv
C:\Users\MIMI\Documents\Internship M2\Matlab scripts\CA_analysis\CalmAn\deconv
C:\Users\MIMI\Documents\Internship M2\Matlab scripts\CA_analysis\CalmAn\deconv
C:\Users\MIMI\Documents\Internship M2\Matlab scripts\CA_analysis\CalmAn\deconv
C:\Users\MIMI\Documents\Internship M2\Matlab scripts\CA_analysis\CalmAn\docs
C:\Users\MIMI\Documents\Internship M2\Matlab scripts\CA_analysis\CalmAn\docs\nc
C:\Users\MIMI\Documents\Internship M2\Matlab scripts\CA_analysis\CalmAn\endosc
C:\Users\MIMI\Documents\Internship M2\Matlab scripts\CA_analysis\CalmAn\endosc
C:\Users\MIMI\Documents\Internship M2\Matlab scripts\CA_analysis\CalmAn\endosc
C:\Users\MIMI\Documents\Internship M2\Matlab scripts\CA_analysis\CalmAn\live scri
C:\Users\MIMI\Documents\Internship M2\Matlab scripts\CA_analysis\CalmAn\tests

It takes the tiff as an input and returns a motion corrected tiff that contains *"MotCorr"* in its name.

Upload this motion corrected movie on your Drive and rename it as "*birthdate_surgerydate_imagingdate_fieldname_MotCorr*", you will need it later for the cell type inference.

⚠ If you observe a shift greater than 10% of the frame or a movie with violent and recurring shifts, you have to annotate those frames and delete them later. If there are too many of them, the best solution is to leave the movie and redo the experiment. Indeed, the relevance of the data analysis is at stake and it will become a waste of time to correct the distortions.

Creation of configuration files

Run the code `nwb_yaml_generator`, located at hippocampal-network-emergence>src>deeptada. This python code will search for both excel files and transform them into panda frames, which will be stored, in .yaml format in the directory yaml_creation>yaml_files. There are 3 configuration files: one for the abf, one for the session data and one for the subject data.

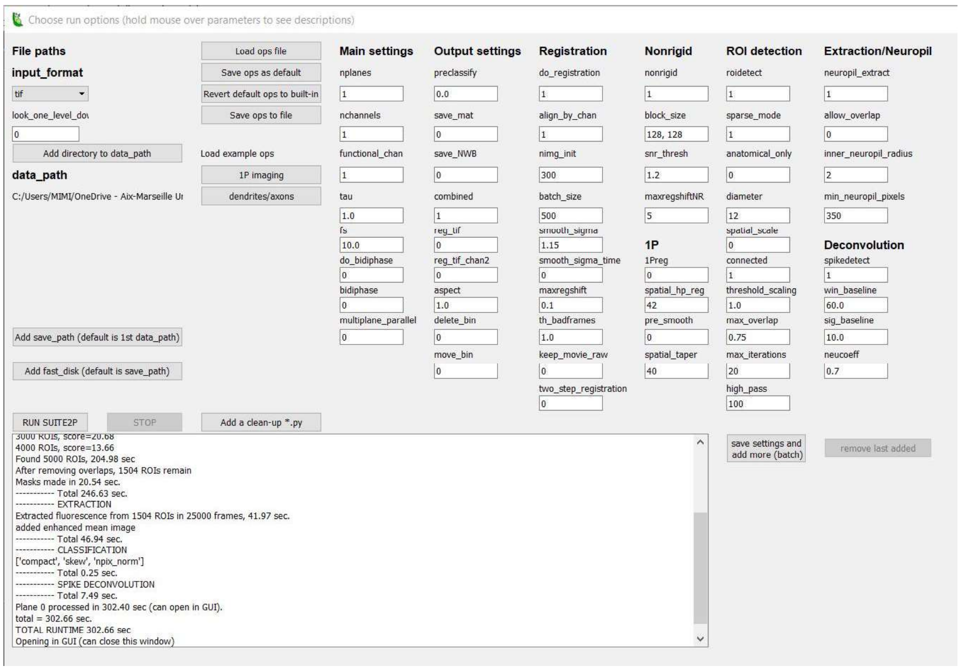
Cell segmentation

The segmentation step is performed by suite2p (that you installed before). Open an anaconda prompt, change the directory to where your file environment.yml is located and type the following commands:

- `conda activate suite2p` will change the environment into a conda one.
- `python -m suite2p` will run the Graphic User Interface (GUI) of suite2p.

The GUI allows to visualize the calcium imaging movies, detect spikes or cells as Regions Of Interest (ROIs).

For the cellular segmentation, you only need to provide the directory where you put your tiffs and launch the segmentation. Select `File > Run suite2p` the "choose run options" window (below) appears. Choose your file directory with `Add directory to data_path` and then `Run SUITE2P`.


Choose run options (hold mouse over parameters to see descriptions)
Load ops file
Save ops as default
Revert default ops to built-in
Save ops to file
Add directory to data_path
Load example ops
1P imaging
dendrites/axons
Add save_path (default is 1st data_path)
Add fast_disk (default is save_path)
RUN SUITE2P
STOP
Add a clean-up *.py
save settings and add more (batch)
remove last added

File paths

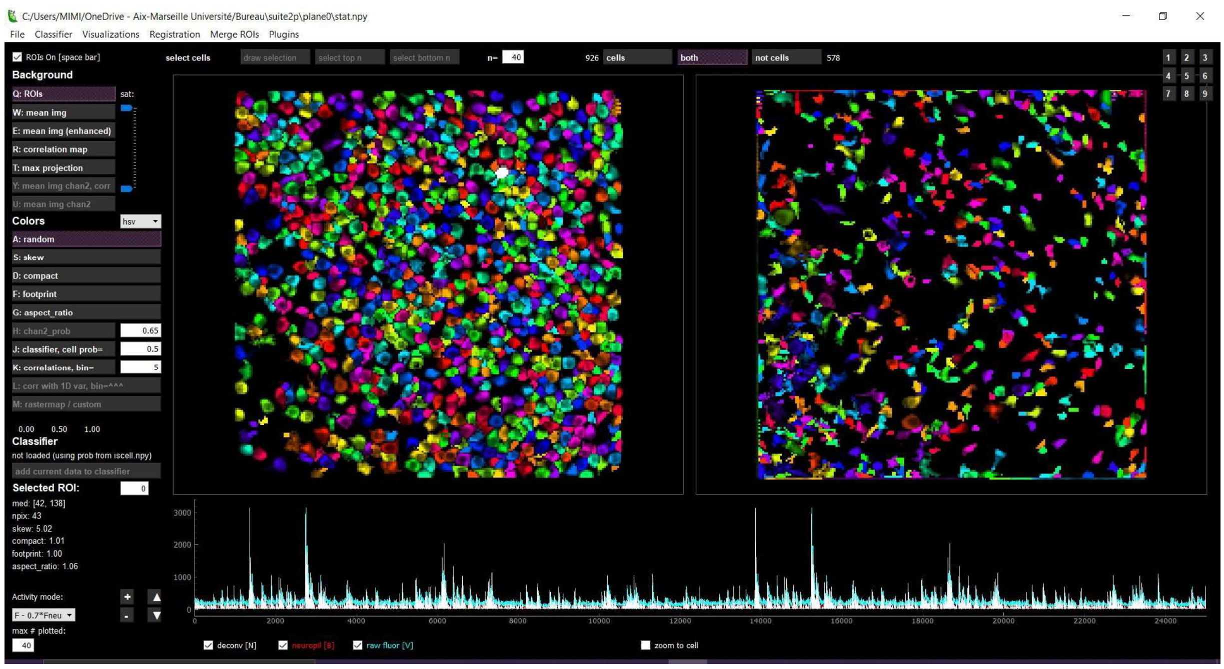
input_format: tif
look_one_level_do: 0

data_path
C:/Users/MIMI/OneDrive - Aix-Marseille Ur

Main settings		Output settings		Registration		Nonrigid		ROI detection		Extraction/Neuropil	
nplanes	1	preclassify	0.0	do_registration	1	nonrigid	1	roidetect	1	neuropil_extract	1
nchannels	1	save_mat	0	align_by_chan	1	block_size	128, 128	sparse_mode	1	allow_overlap	0
functional_chan	1	save_NWB	0	nimg_init	300	snr_thresh	1.2	anatomical_only	0	inner_neuropil_radius	2
tau	1.0	combined	1	batch_size	500	maxregshiftNR	5	diameter	12	min_neuropil_pixels	350
fs	10.0	reg_tif	0	smooth_sigma	1.15	**1P**		spatial_scale	0	**Deconvolution**	
do_bidiphase	0	reg_tif_chan2	0	smooth_sigma_time	0	1Preg	0	connected	1	spikedetect	1
bidiphase	0	aspect	1.0	maxregshift	0.1	spatial_hp_reg	42	threshold_scaling	1.0	win_baseline	60.0
multiplane_parallel	0	delete_bin	0	th_badframes	1.0	pre_smooth	0	max_overlap	0.75	sig_baseline	10.0
		move_bin	0	keep_movie_raw	0	spatial_taper	40	max_iterations	20	neucoeff	0.7
				two_step_registration	0			high_pass	100		

```
3000 ROIs, score=20.68
4000 ROIs, score=13.66
Found 5000 ROIs, 204.98 sec
After removing overlaps, 1504 ROIs remain
Masks made in 20.54 sec.
----------- Total 246.63 sec.
----------- EXTRACTION
Extracted fluorescence from 1504 ROIs in 25000 frames, 41.97 sec.
added enhanced mean image
----------- Total 46.94 sec.
----------- CLASSIFICATION
['compact', 'skew', 'npix_norm']
----------- Total 0.25 sec.
----------- SPIKE DECONVOLUTION
----------- Total 7.49 sec.
Plane 0 processed in 302.40 sec (can open in GUI).
total = 302.66 sec.
TOTAL RUNTIME 302.66 sec
Opening in GUI (can close this window)
```

The segmentation is achieved when the command "opening in GUI (can close this window)" appears. Close the window and you will find the main window with the segmentations results.



At this point, you need to adjust "by hand" the cells detected. For that, choose both options, you will obtain two frames. The left is for the cells detected as neurons by suite2p and the right frame is for those who are not neurons. To select a cell and change its class (neurons/not neurons), left-click + right-click on the highlighted neurons and it's done! If you are using a touchpad, selecting the cell and using both fingers will do the trick.

After the manual segmentation, the output is saved in the same directory as your tiffs. It's a file named suite2p with 6 .npy files and a binary (.bin) file. It gathers the outlines and the class [cell/not cell] for each outline detected (spks.npy and iscell.npy), the individual activity (F.npy : fluorescence traces and Fneu.npy: Neuropil fluorescence traces) and the statistics for each cell (stat.npy).

Prediction of the cellular type

For both classifiers, you will run them on the jupyter notebook. First you will run the cell type classifier that you will find on the gitlab (if available), if you don't find it, ask Robin D. or Julien D. .

If you are using the classifier for the first time, you need to upload and create some files:

- First, create a folder in your Google drive named **‘CI_data’**. In **CI_data**, create two folders **‘data’** and **‘model’**.
- In **‘model’**, upload the model and the weights from each cellular type classifier (in .h5 and .json format): you will find them on the gitlab (if available) or with Robin D.
- In “**data**”, create a “**suite2p_*id_mouse***” file containing all the .npy files from the segmentation (stat, spks, ops, iscell, Fneu and F).
- In “**data**” again, copy the file “**cell_type_yaml_files**” containing the “**pyr_vs_ins_vs_noise_multi_class.yaml**” file.

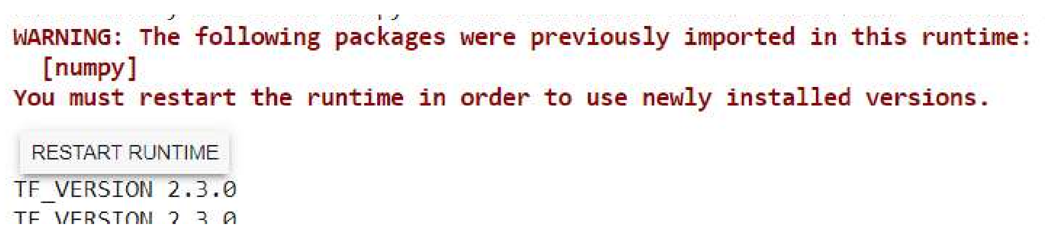
Before launching the predictions, you need to modify a few lines in the code. First, modify the directory to where you put your motion corrected calcium imaging movie:

```
# root path, just used to avoid copying the path everywhere
root_path = 'gdrive/My Drive/CI_data/'
```

After that, change the identifier into the name of your experiment and the age of the pup.

```
# ------------------------------------------------------------ #
# ------------------------------------------------------------ #
identifier = "210128_210206_1_210206_a001"
# ------------------------------------------------------------ #
# ------------------------------------------------------------ #
age = 9
```

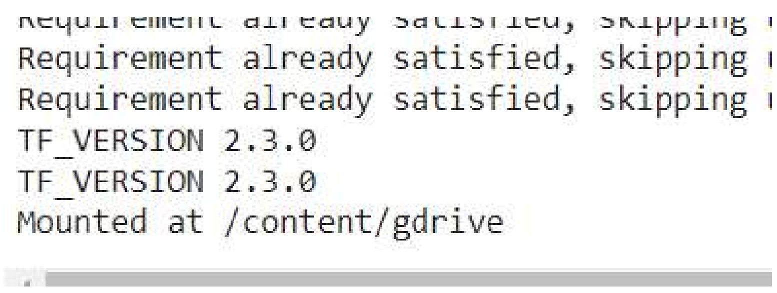
After the modifications required above, run the first cell by hitting the play button. You will have to ‘**RESTART RUNTIME**’



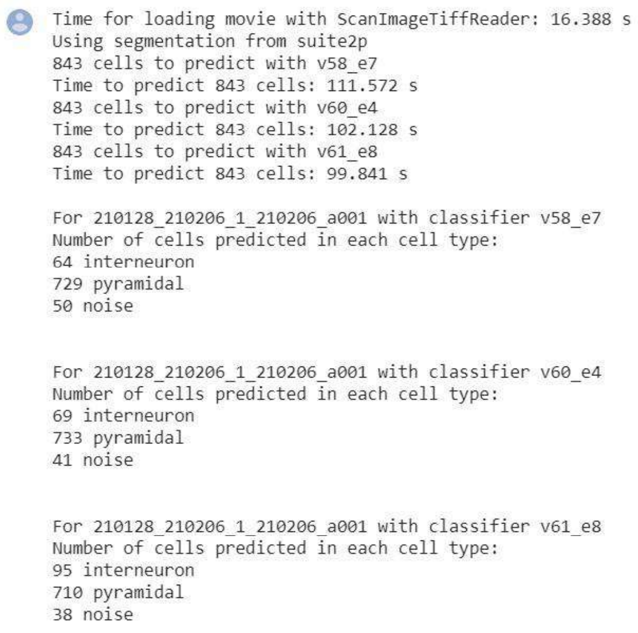
```
WARNING: The following packages were previously imported in this runtime:
  [numpy]
You must restart the runtime in order to use newly installed versions.
```

```
TF_VERSION 2.3.0
TF_VERSION 2.3.0
```

and rerun the first cell, at this step, click on the link to retrieve the password. At the end of the re-run, this output will be displayed:


```
Requirement already satisfied, skipping
Requirement already satisfied, skipping
Requirement already satisfied, skipping
TF_VERSION 2.3.0
TF_VERSION 2.3.0
Mounted at /content/gdrive
```

Now you can run the second cell and begin the predictions. At the end of the session, you will have an output similar to the picture below:


```
Time for loading movie with ScanImageTiffReader: 16.388 s
Using segmentation from suite2p
843 cells to predict with v58_e7
Time to predict 843 cells: 111.572 s
843 cells to predict with v60_e4
Time to predict 843 cells: 102.128 s
843 cells to predict with v61_e8
Time to predict 843 cells: 99.841 s

For 210128_210206_1_210206_a001 with classifier v58_e7
Number of cells predicted in each cell type:
64 interneuron
729 pyramidal
50 noise


For 210128_210206_1_210206_a001 with classifier v60_e4
Number of cells predicted in each cell type:
69 interneuron
733 pyramidal
41 noise


For 210128_210206_1_210206_a001 with classifier v61_e8
Number of cells predicted in each cell type:
95 interneuron
710 pyramidal
38 noise
```

The output of one cellular type classifier is a dictionary where the keys are the 3 cellular types, namely interneuron, pyramidal and noisy cell, and their associated probabilities, computed by Deepcinac. Retrieve the 3.npz output files in “**results**”. Download them and keep in mind their directories, you will need them later.

Note that, it is important to make the cell type inference before the prediction of the activity, since the inference of the activity depends on the cellular type. Therefore, you must run the cell type classifier first and the 3 different activity classifiers (one for each cellular type) after.

Organize the nwb content (1st time)

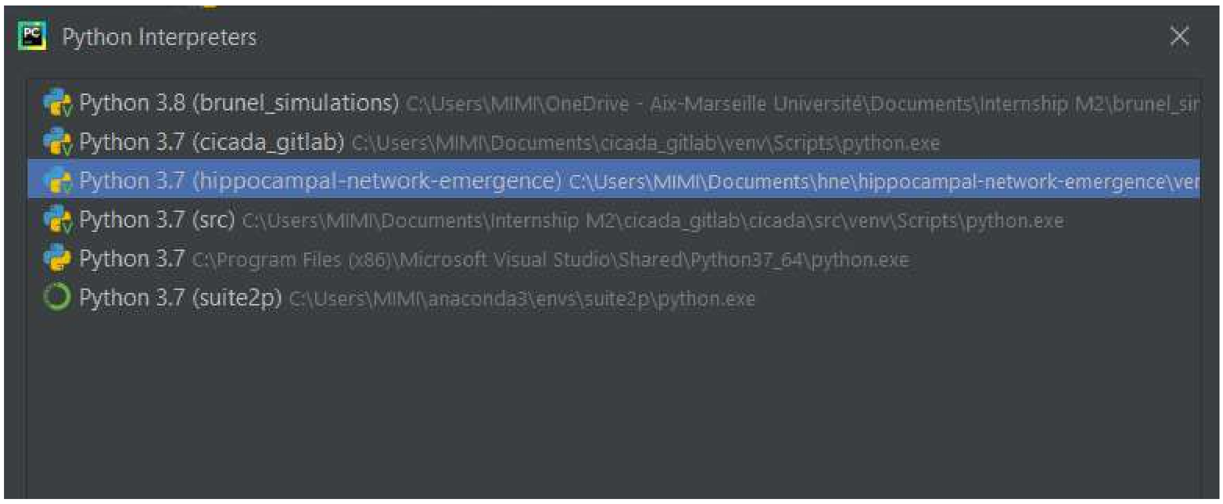
After the cellular type inference, download the 3 .npz files in your local folder, with the motion corrected and original movie. Now, you need to organize your data in subfolders before inferring the neuronal activity.

This is performed in multiple steps. First, you need the python codes from the **hne** and **deepcinac** repos that you will clone.

Warning: Before cloning the repositories, beware that you will need a computer with a GPU such that it is CUDA-enabled (check the tables on Nvidia website)

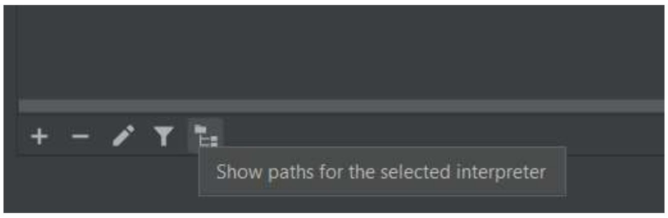
You will find the hne repos on github https://github.com/pappyhammer/hippocampal-network-emergence**.** As the other repos., create a file on your disk named "hne" and clone the repos there. Create a new project by selecting the "hippocampal-network-emergence" file and add the python interpreter path by going in Settings>hippocampal-network-emergence, click on the setting button>Add... and create the new interpreter.

Hit the settings button again, select Show All and it should display something similar :

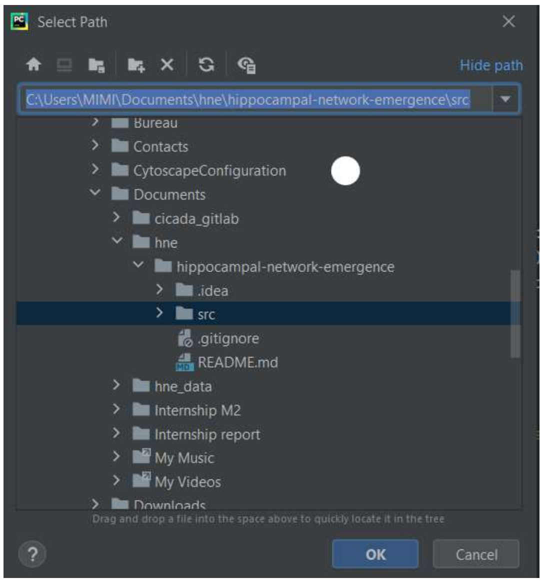


```
Python 3.8 (brunel_simulations) C:\Users\MIMI\OneDrive - Aix-Marseille Université\Documents\Internship M2\brunel_sir
Python 3.7 (cicada_gitlab) C:\Users\MIMI\Documents\cicada_gitlab\venv\Scripts\python.exe
Python 3.7 (hippocampal-network-emergence) C:\Users\MIMI\Documents\hne\hippocampal-network-emergence\ver
Python 3.7 (src) C:\Users\MIMI\Documents\Internship M2\cicada_gitlab\cicada\src\venv\Scripts\python.exe
Python 3.7 C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python37_64\python.exe
Python 3.7 (suite2p) C:\Users\MIMI\anaconda3\envs\suite2p\python.exe
```

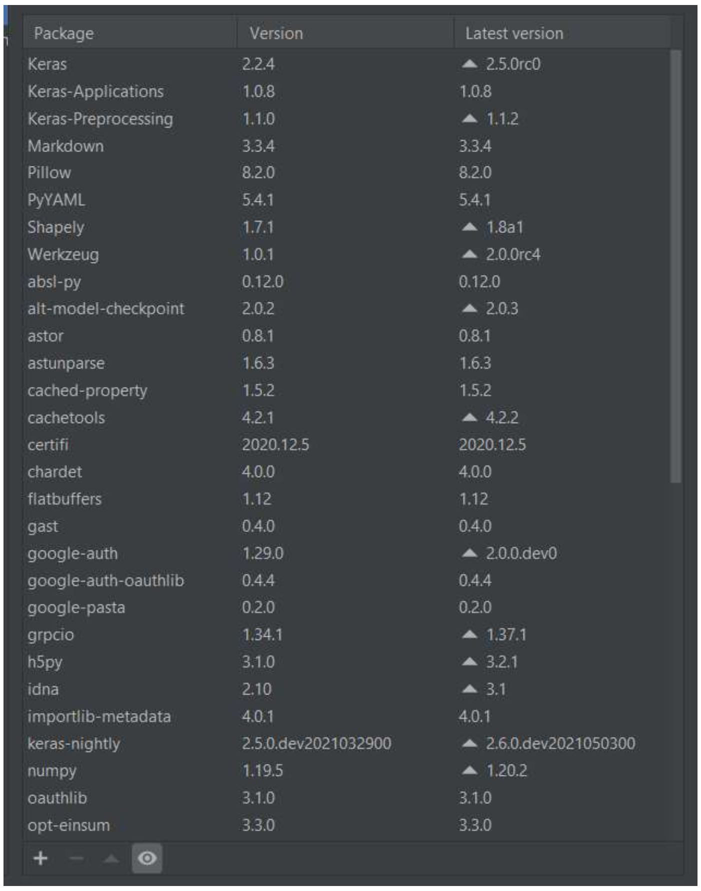
Here you have all the added python interpreter paths.



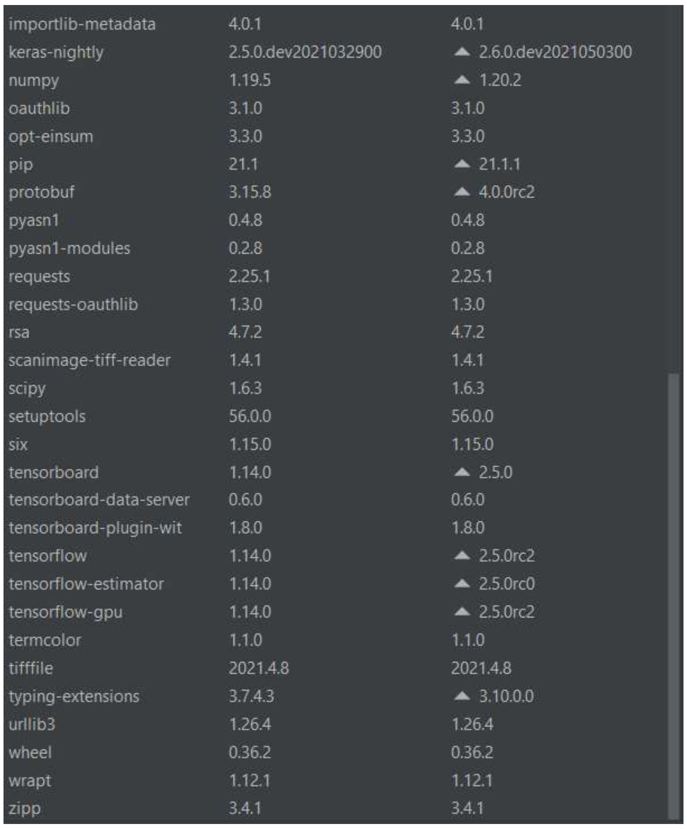
Select the Python interpreter for **hne** and click on the button bottom-right, it will display the interpreters' paths. Those are the allowed directories given to the interpreter. Use the **+** button to add the **src** file as below. This step is very important, it will allow the access of the interpreters to all the codes in this **src** file.



For the deepcinac repos, clone it at https://gitlab.com/cossartlab/deepcinac and put it in the "**deepcinac_gitlab**" file. As before add the corresponding interpreter and select the path deepcinac_gitlab>deepcinac>src to add **src** to the interpreter's path. For **Deepcinac**, you need to download some Python modules. As you did with **Cicada**, go in File>Settings, choose the interpreter for **hne** and download the modules below (see two screenshots below). Most of them are from the final version, except for tensorflow-gpu (v.1.14.0), tensorflow (v.1.14.0) and keras (v.2.2.4).

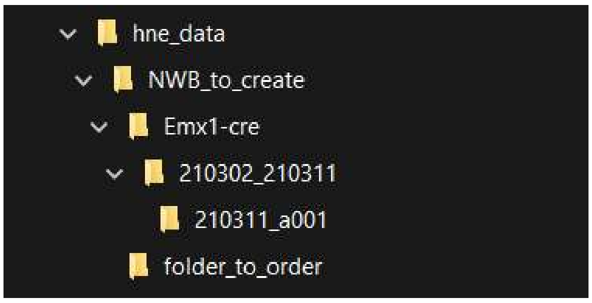


Package	Version	Latest version
Keras	2.2.4	▲ 2.5.0rc0
Keras-Applications	1.0.8	1.0.8
Keras-Preprocessing	1.1.0	▲ 1.1.2
Markdown	3.3.4	3.3.4
Pillow	8.2.0	8.2.0
PyYAML	5.4.1	5.4.1
Shapely	1.7.1	▲ 1.8a1
Werkzeug	1.0.1	▲ 2.0.0rc4
absl-py	0.12.0	0.12.0
alt-model-checkpoint	2.0.2	▲ 2.0.3
astor	0.8.1	0.8.1
astunparse	1.6.3	1.6.3
cached-property	1.5.2	1.5.2
cachetools	4.2.1	▲ 4.2.2
certifi	2020.12.5	2020.12.5
chardet	4.0.0	4.0.0
flatbuffers	1.12	1.12
gast	0.4.0	0.4.0
google-auth	1.29.0	▲ 2.0.0.dev0
google-auth-oauthlib	0.4.4	0.4.4
google-pasta	0.2.0	0.2.0
grpcio	1.34.1	▲ 1.37.1
h5py	3.1.0	▲ 3.2.1
idna	2.10	▲ 3.1
importlib-metadata	4.0.1	4.0.1
keras-nightly	2.5.0.dev2021032900	▲ 2.6.0.dev2021050300
numpy	1.19.5	▲ 1.20.2
oauthlib	3.1.0	3.1.0
opt-einsum	3.3.0	3.3.0

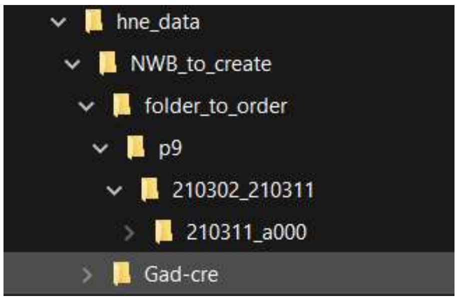


importlib-metadata	4.0.1	4.0.1
keras-nightly	2.5.0.dev2021032900	▲ 2.6.0.dev2021050300
numpy	1.19.5	▲ 1.20.2
oauthlib	3.1.0	3.1.0
opt-einsum	3.3.0	3.3.0
pip	21.1	▲ 21.1.1
protobuf	3.15.8	▲ 4.0.0rc2
pyasn1	0.4.8	0.4.8
pyasn1-modules	0.2.8	0.2.8
requests	2.25.1	2.25.1
requests-oauthlib	1.3.0	1.3.0
rsa	4.7.2	4.7.2
scanimage-tiff-reader	1.4.1	1.4.1
scipy	1.6.3	1.6.3
setuptools	56.0.0	56.0.0
six	1.15.0	1.15.0
tensorboard	1.14.0	▲ 2.5.0
tensorboard-data-server	0.6.0	0.6.0
tensorboard-plugin-wit	1.8.0	1.8.0
tensorflow	1.14.0	▲ 2.5.0rc2
tensorflow-estimator	1.14.0	▲ 2.5.0rc0
tensorflow-gpu	1.14.0	▲ 2.5.0rc2
termcolor	1.1.0	1.1.0
tifffile	2021.4.8	2021.4.8
typing-extensions	3.7.4.3	▲ 3.10.0.0
urllib3	1.26.4	1.26.4
wheel	0.36.2	0.36.2
wrapt	1.12.1	1.12.1
zipp	3.4.1	3.4.1

For the creation of nwb, you will follow this architecture (you can change the name of the transgenic line but make sure to adapt it to the code):



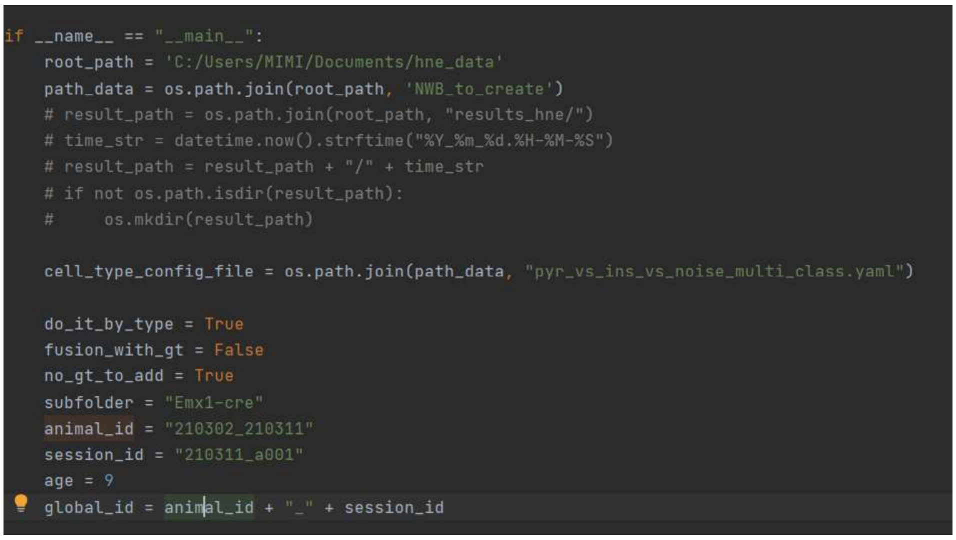
```
hne_data
  NWB_to_create
    Emx1-cre
      210302_210311
        210311_a001
    folder_to_order
```



```
hne_data
  NWB_to_create
    folder_to_order
      p9
        210302_210311
          210311_a000
    Gad-cre
```

Make sure to have 3 levels in ‘**folder_to_order**’, for instance, here the age is the first, then the birthdate_surgerydate the second level and the final one is the imagingdate_fov.

Before reorganizing the nwb content, modify the root_path, the sub_folder and the other parameters in the *cell_type_predictions_fusion.py*, *organize_data_dir_for_nwb.py* and *param_hne.txt* based on the name of your folders:

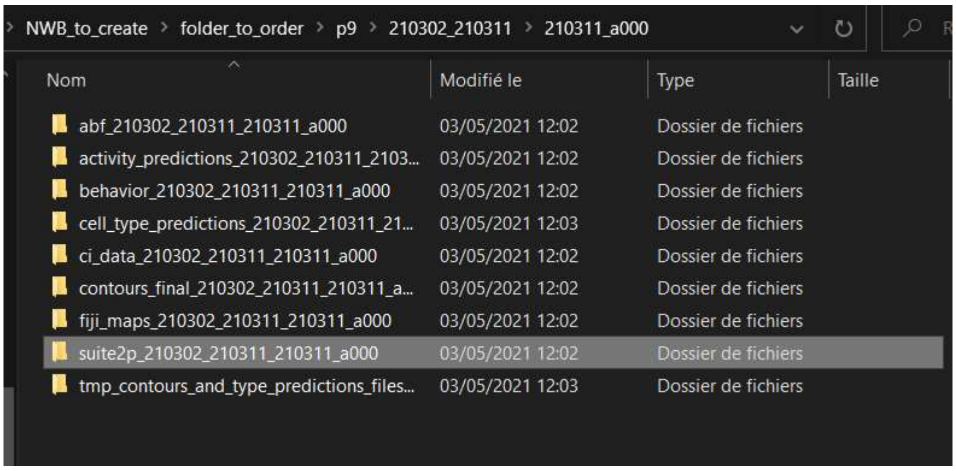


```
if __name__ == "__main__":
    root_path = 'C:/Users/MIMI/Documents/hne_data'
    path_data = os.path.join(root_path, 'NWB_to_create')
    # result_path = os.path.join(root_path, "results_hne/")
    # time_str = datetime.now().strftime("%Y_%m_%d.%H-%M-%S")
    # result_path = result_path + "/" + time_str
    # if not os.path.isdir(result_path):
    #     os.mkdir(result_path)

    cell_type_config_file = os.path.join(path_data, "pyr_vs_ins_vs_noise_multi_class.yaml")

    do_it_by_type = True
    fusion_with_gt = False
    no_gt_to_add = True
    subfolder = "Emx1-cre"
    animal_id = "210302_210311"
    session_id = "210311_a001"
    age = 9
    global_id = animal_id + "_" + session_id
```

Gather the files and folders in “**folder_to_order**”. You will need the 3 configuration files in .yaml, the motion corrected movie in .tiff format (make sure it is named correctly ie id_MotCorr.tif), the original movie in .tiff (named correctly), the .abf file, the suite2p folder (with plane0 folder content at the same level as the run file), the behavior monitoring movies and the manual annotation.

The files will be reorganized automatically by using the script `organize_data_dir_for_nwb`. First, add a new configuration in Pycharm : for that, click on the drop menu upper-right next to the play button. Click on Edit config… . Click on the upper left + icon to add a python configuration. Name it as you want, select the directory of the Python script you want to run, select the project linked to the script and verify the Python interpreter and the working directory. Once the configuration is settled, you can run the script.

At the end of this step, data is organized in subfolders (see figure below):



› NWB_to_create › folder_to_order › p9 › 210302_210311 › 210311_a000

Nom	Modifié le	Type	Taille
abf_210302_210311_210311_a000	03/05/2021 12:02	Dossier de fichiers	
activity_predictions_210302_210311_2103...	03/05/2021 12:02	Dossier de fichiers	
behavior_210302_210311_210311_a000	03/05/2021 12:02	Dossier de fichiers	
cell_type_predictions_210302_210311_21...	03/05/2021 12:03	Dossier de fichiers	
ci_data_210302_210311_210311_a000	03/05/2021 12:02	Dossier de fichiers	
contours_final_210302_210311_210311_a...	03/05/2021 12:02	Dossier de fichiers	
fiji_maps_210302_210311_210311_a000	03/05/2021 12:02	Dossier de fichiers	
suite2p_210302_210311_210311_a000	03/05/2021 12:02	Dossier de fichiers	
tmp_contours_and_type_predictions_files...	03/05/2021 12:03	Dossier de fichiers	

Move it in the corresponding folder `Nwb_to_create > Mouse_Type > dateofbirth_surgerydate > imagingdate_a000` and also move the 3 .npz in the cell_type_predictions folders to the tmp_contours_and_type_predictions one.

The second step consists of fusing the cell type predictions. For that, locate the script **cell_type_predictions_fusion.py** at hne>src>cell_type_predictions_fusion.py , modify the path to `Nwb_to_create > Mouse_Type > dateofbirth_surgerydate > imagingdate_a000` with all the subfolders organized and run it. The script will save the **cell_type_predictions_fusion.npy** file in the cell_type_predictions folder. Upload it in your Drive, in CI_data>data .

Activity inference

List of folders needed

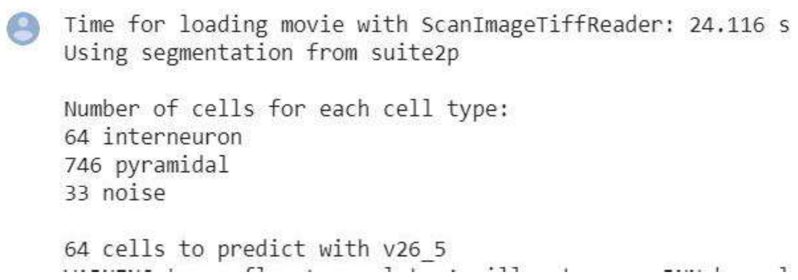
First, find the classifier script on the gitlab of Deepcinac (if available), otherwise, ask Robin D. or Julien D.. Again as before, do the same modifications on the script of the activity classifier.

Before inferring the neuronal activity, you will find below a list of files to gather and upload on your drive:

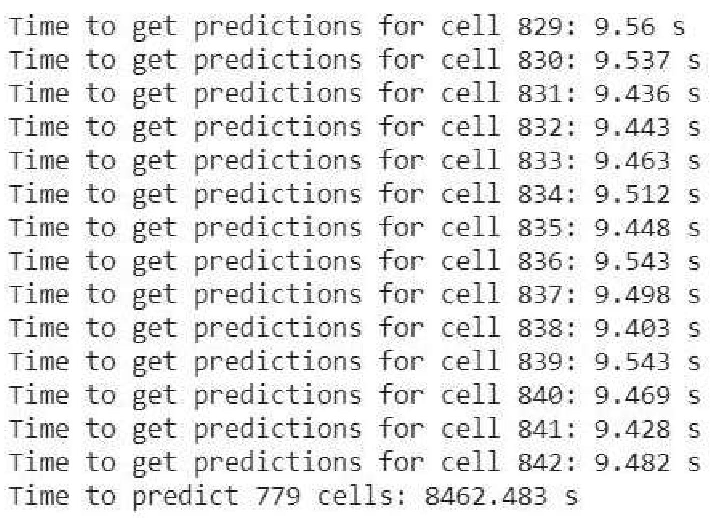
The models and weights from the activity classifier for each cell type (you will find it on gitlab, if available or please ask Robin D.), the **pyr_vs_ins_vs_noise_multi_class.yaml** as above, the motion corrected movie in .tiff, the .npy file gathering the new contours made on suite2p and the .npy file returned from the **cell_type_predictions_fusion.py** script**.**

Neuronal activity inference

After that, run the script as with the activity classifier (*i.e.* do not forget to ***rerun startime*** and copy the password). The execution is longer than for the cellular type classifier (count forty minutes to more than one hour) therefore, make sure not launch it at the last minute. The output of the script will be similar to the screenshot below:



```
Time for loading movie with ScanImageTiffReader: 24.116 s
Using segmentation from suite2p

Number of cells for each cell type:
64 interneuron
746 pyramidal
33 noise

64 cells to predict with v26_5
```

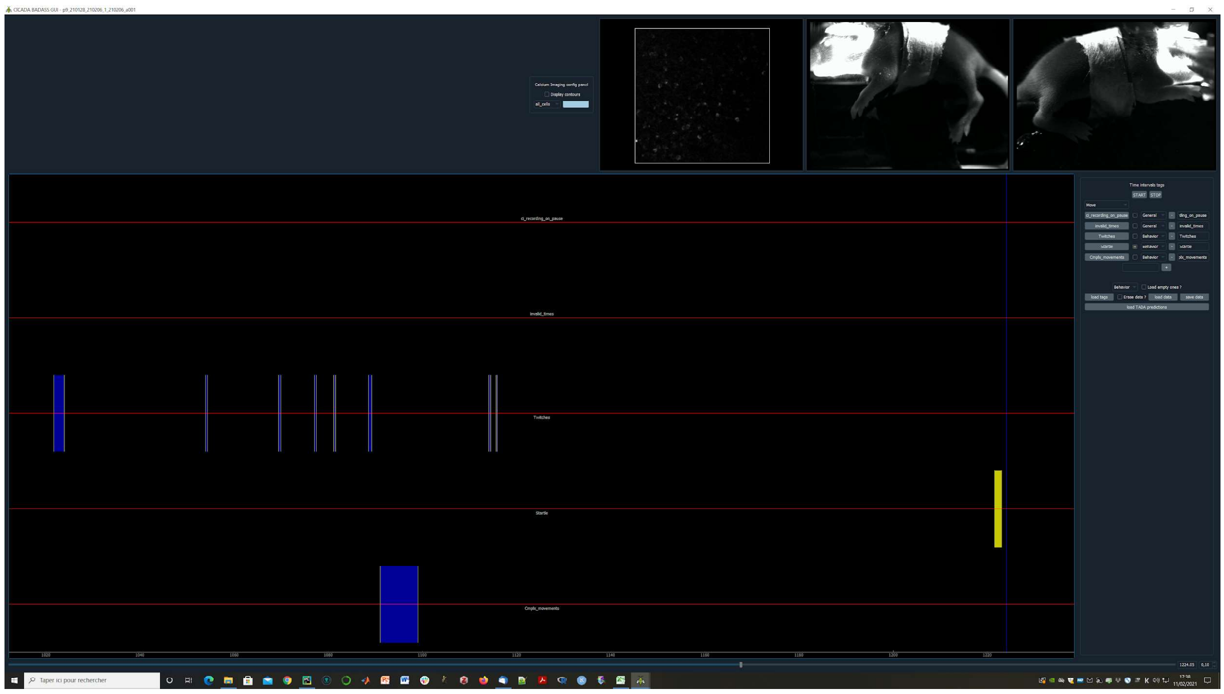


```
Time to get predictions for cell 829: 9.56 s
Time to get predictions for cell 830: 9.537 s
Time to get predictions for cell 831: 9.436 s
Time to get predictions for cell 832: 9.443 s
Time to get predictions for cell 833: 9.463 s
Time to get predictions for cell 834: 9.512 s
Time to get predictions for cell 835: 9.448 s
Time to get predictions for cell 836: 9.543 s
Time to get predictions for cell 837: 9.498 s
Time to get predictions for cell 838: 9.403 s
Time to get predictions for cell 839: 9.543 s
Time to get predictions for cell 840: 9.469 s
Time to get predictions for cell 841: 9.428 s
Time to get predictions for cell 842: 9.482 s
Time to predict 779 cells: 8462.483 s
```

The output of the script is kept in a dictionary as well, with `Deepcinac Recording identifiers` as keys and the predictions for each cell as a value.

The annotation of the behavior

This step is performed with the Badass GUI of CICADA. The GUI is designed to perform polyvalent observations. Indeed, it allows to annotate other observations than the behavior and movement: for example, the behavior monitoring (*i.e.* signals of both cameras) can be replaced with electrophysiological signals.



Once the Badass GUI is launched, select the results directory: the behavior subfolder; then, open the two behavior monitoring movies (by clicking on the two frames on the right). On the right panel, create the tags associated with the respective observations, for example: for behavior annotation, you can create “twitche”, “startle”, “complex movements”. On the left frame, you will see multiple channels, corresponding to each tag, these channels are designed to annotate each observation (associated to the tag) for a certain period of time. To annotate the event, select the tag and simply drag it towards the respective channel, then adjust the size of the tag so that it meets the start and the end of the observation.

The GUI takes as inputs the behavior monitoring movies and the motion corrected imaging movie in .tiff format. When the annotation is done, the results are saved in a “behavior file” in .npz format. The .npz file is a zip archive which contains files (in our case, a dictionary). The files in the example above correspond to the tags and each file contains a d-array. The d-array is a N dimensional array in Python, in our case, the d-array is in 2D, the first line is the start time and the second line is the end time of the observation.

Creation of nwb files (final version)

The .nwb file is like a lab notebook, it organizes the data in an understandable way to humans and programmatic interpretation, besides, the format is designed to be usable for software tools and analysis scripts (for instance, CICADA).

After the first organization, your data should be ordered in many subfolders. Run the **`nwb_pre_processing.py`** with an adapted configuration that you will create through the same process as earlier. [Don't forget the list of files for the content of the nwb and where they should be located].

Note that if your experiment doesn't include behavior files, the creation of the nwb file is not impacted, so does the data analysis. However, nwb files cannot be modified, therefore, before creating nwb files, make sure you do not want to modify something in the preprocess, for example, the metadata content or the tags on the behavior files.

Data analysis

After the creation of the nwb file, you can perform many tasks, from the basic analysis to an advanced analysis. These tasks are well performed on CICADA. You can compute the frequency, the amplitude of the transients. You can also display the raster plots or the cellular-dependent activity from your calcium imaging movies.